Oscillation-free video adaptation at application layer on server side and experiments using DCCP

Wassim Ramadan, Eugen Dedu, Julien Bourgeois

Oscillation-free video adaptation at application layer on server side and experiments using DCCP

Wassim Ramadan, Eugen Dedu and Julien Bourgeois

UFC/FEMTO-ST Institute, UMR CNRS 6174, 4 pl. Tharradin, 25200 Montbéliard, France
Email: Eugen.Dedu@pu-pm.univ-fcomte.fr

Nowadays, video data transfers account for much of the Internet bandwidth and a huge number of users use it daily. However, despite its apparent interest, video streaming is still done in a suboptimal manner. Indeed, more and more high definition and high quality videos are nowadays stored on Internet but they are not accessible for everybody because a high and stable bandwidth is needed to stream them; also, during video conferencing, the highest possible quality often exceeds the available bandwidth. Hence, a lower bitrate encoding is usually chosen but it leads to lower quality and network under-utilisation too. This paper presents VAAL, a simple and efficient method designed to use optimally network resources and to ameliorate user video experience. It involves only the application layer on the server. The main idea of VAAL is that it checks TCP-friendly transport protocol buffer overflows and adapts the video bitrate accordingly; as a result, the bitrate constantly matches the network bandwidth. It can be used together with ZAAL, a novel algorithm aiming to avoid quality oscillations. Experimental results show that the video adaptation using VAAL+ZAAL performs much better compared to the currently widely-used static encoding, making it a strong candidate for hard real-time video streaming.

Keywords: Video streaming, Content adaptation, DCCP, Rate control, TCP-friendliness

1. INTRODUCTION

In recent years, the number of videos encoded in several bitrates has significantly increased to the point that they become accessible to everybody. Also, the quality of a video conference can greatly vary, depending on the encoding quality chosen by the sender. These videos are delivered to final users using streaming services. Multimedia streaming services over Internet, as well as the demand for higher quality from final clients are in constant progression. New video standards like HD and 3D are demanding for more bandwidth. Available bandwidth variation has also to be taken into account so that buffering time can be shortened. Watching such videos on a network with unstable bandwidth is not a comfortable experience, especially if the bandwidth becomes smaller than the bitrate. For example, wireless networks use various network technologies with different characteristics, and they can change over time (interferences, mobility etc.) Another example is networks with shared bandwidth among several users, which could make the available bandwidth unstable.

Additionally, more and more network applications, for example real-time media like audio and video streaming, can accept a certain level of losses. If they use TCP (Transmission Control Protocol), they have to pay the price for full reliability, with great latency. On the other hand, UDP (User Datagram Protocol) lacks congestion avoidance support. RTP (Real-time Transport Protocol) [1], while being a widely-used protocol for multimedia streaming, is an application protocol; as such, it is put on top of a transport protocol, such as TCP or UDP, hence it does not cope with transport protocol problems.

Another promising protocol for these applications is DCCP (Datagram Congestion Control Protocol), recently standardised as RFC4340 [2]. It can be seen as TCP minus reliability and in-order delivery of packets, two key points in video streaming, or as UDP plus congestion control. For our purposes, two interesting points of DCCP are that it allows choosing the congestion control used during communication and that it uses acknowledgements. Among the currently three standardised congestion control protocols, TFRC (TCP-Friendly Rate Control) is the most adapted to video streaming [3]. Also, acknowledgement packets give useful information to the sender, such as the lost packets and ECN (Explicit Congestion Notification) marks.

For the above protocols, especially DCCP and TCP, video transmission is controlled at the network layer and the application is not involved at all. For video streaming in a network with highly variable bandwidth during a connection, an adaptation of the video to the network characteristics is very important. A cooperative approach between application layer and network layer can improve the video quality perceptible by the final user.

The adaptation method we propose (VAAL, Video Adaptation at Application Layer) uses transport protocol buffer overflow as a solution to find out the available bandwidth and to adapt the video content bitrate to the discovered bandwidth. Each n fixed seconds, the server application computes the number of packets which failed to be written to the socket buffer. This number is used to control the video bitrate afterwards. A high number means smaller bandwidth and smaller bitrate. Zero error indicates either a stable or a greater bandwidth, so the bitrate of sent video could be increased. In this way, the bitrate of the watched video is fixed during each period of n seconds, and can change only between periods.

The above adaptation, known in the literature as "rate adaptive video control", can be done by controlling some other video parameters, such as number of frames per second (FPS) and image size.

The three parameters presented above allow to optimise the *application* part of the video streaming. For yet better results, these methods could be coupled with other methods. For example, a well-known problem with losses in wireless networks is that they cannot be differentiated from congestion losses, hence the sender reduces the throughput while it should not [4]. Another optimisation on the *network part* useful on lossy links is the FEC (Forward Error Correction) ([5] for example). Also, *video-specific network techniques* allow for example to prioritise [6] or retransmit [7] only important packets (I packets in an MPEG-encoded video) on the server side. A commonly-used such technique is ALF (Application-Level Framing), which cuts intelligently video data in packets (avoiding for example to put one small video frame in two packets, which would be more sensible to packet loss); this is to be used in conjunction with the MTU (Maximum Transmission Unit) of the network.

It is important to understand that our method is only a small part in the optimisation chain for video streaming, which can be used and is beneficial independently of the other methods presented above. As such, *we consider in this article only network parameters and network performance criteria.*

The rate control is not a new idea to ameliorate video transmission. In the literature, there are a multitude of papers about this topic. Most of them were written 5–10 years ago, where solutions using new/modified transport protocols were used (such as [8]), which *are not deployable in reality* and risky from the congestion control point of view. Moreover, only few of them use experiments. The only papers we have found about rate control over DCCP are [9, 10], and they use simulations[1]. A new trend in adaptive video is also to use HTTP over TCP, such as the recent ISO MPEG DASH standard [11], which has advantages, but also drawbacks, such as, in videoconferencing, mandatory 100% reliability given by TCP use.

In this context, this is the first time that the *buffer overflow method is analysed for video adaptation.* Moreover, our paper is, to our knowledge, *the first paper which uses DCCP in real experiments of video streaming in wireless networks.* Our solution is *very simple to deploy*, as only the application on the sender side needs to be modified (does not need to modify the receiver, nor the transport protocol). As a corollary, our method works with any transport protocol which has a congestion control.

The scope of our method is unidirectional and bidirectional communications, such as VoD (Video on Demand) and videoconferencing. Our method applies ideally to videoconferencing, because it is delay-sensitive and it consists mainly in a few simple additions/divisions per second and updating the value of the quantisation parameter; also, there is no need of support, such as caches, from CDN (Content Delivery Networks).

This paper is organized as follows. The motivation of this paper is given in section 2. Section 3 presents related work for video adaptation. Section 4 presents our VAAL method and its implementation on GNU/Linux, and section 5 presents ZAAL, a novel algorithm to avoid quality oscillations in a video adaptation method. Performance of VAAL+ZAAL is evaluated through real experiments in section 6. Finally, section 7 concludes this article and presents some perspectives.

2. MOTIVATIONS

2.1. Advantages of video adaptation over static encoding

Video adaptation is useful when the available bandwidth varies during transmission. This appears in two cases: either the network bandwidth itself changes, or the bandwidth available for the flow changes.

The first case appears in wireless networks, where data rate changes according to radio link quality, and the bandwidth varies often. Data rate decreasing has multiple reasons:

1. interferences due to environment or to presence of another equipment working on the same range of

[1] There are serious concerns about simulation results, such as "around 50% of the papers appeared to be... bogus" and "who has ever validated NS2 code?" (from September 2005 archives of the e2e-interest mailing list). This is especially true for wireless link simulation.

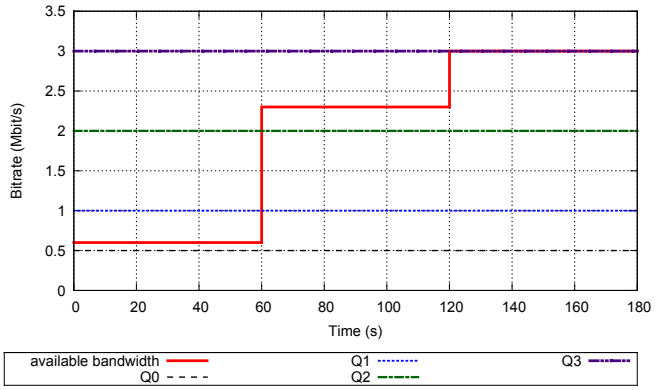


FIGURE 1. Example of available bandwidth vs available bitrates.

frequencies, which decrease the signal to noise ratio for a short period;

2. mobility which, depending on the distance between the mobile and the access point, leads to signal attenuation and might also cause dynamic rate scaling.

This case also appears when an ISP (Internet Service Provider) changes dynamically the bandwidth allocated to a user. Indeed, with the increasing number of streaming services on Internet, more and more traffic is generated, which leads ISPs to limit user bandwidth even during a transfer. We simulate this case by using a traffic shaping model as described in experiments section.

The second case is when a variable number of flows share the same path. The bandwidth available for video streaming is also variable for each flow. We illustrate this case by using a different number of concurrent flows as described in our experiments section.

For these cases, the adaptation helps to take advantage of all the available bandwidth or to avoid numerous lost packets.

2.2. Case study

Classical video transmission uses the same bitrate from the beginning to the end of the transmission. For comparison, we give some theoretical results based on a bandwidth between sender and receiver which changes according to figure 1. It simulates the limits imposed by ISPs, but also (on a larger scale) the dynamics of the available bandwidth between two hosts on a network. As already known, many factors contribute to this dynamics, for example when a user goes further or nearer the access point in a wireless network, or when more or less packets are injected into the network. The bandwidth changing pose problems to the video transmission because when bitrate is smaller than bandwidth, the network is underutilised, and when bitrate is higher than bandwidth, packets will be lost.

The figure presents the available bitrates of a hypothetical video: 512kb/s (Q0), 1Mb/s (Q1), 2Mb/s (Q2) and 3Mb/s (Q3). Also, as shown in the figure, during the 180 seconds of the video transmission the bandwidth changes three times: 600kb/s for the first minute, where 512kb/s video bitrate is the best choice; 2300kb/s for the second minute, where only quality of 3Mb/s should not be used; and 3Mb/s for the last minute, where any video quality could be used.

Quality	*Sent pkts*	*Received pkts*	*Lost pkts*
Q3	69120	45312	34.4%
Q2	46080	35328	23.3%
Q1	23040	19986	13.3%
Q0	11520	11520	0%
Ideal	42240	42240	0%

TABLE 1. Number of sent and received packets for each static bitrate and the ideal case.

In this example it can be noticed that none of the four available bitrates is suited for the whole transmission:

1. Q0 is good for the first minute but choosing it will prevent the user from fully utilising the bandwidth for the remaining time.
2. Q1 is not good for the first minute and has the same drawback as 512kb/s bitrate for the remaining time.
3. Q2 is not good for the first minute; it could be chosen for the second minute, and has the same drawback as 512kb/s bitrate for the remaining time.
4. Finally, Q3 is not good for the first and second minutes; it could be chosen for last minute.

In order to compare the qualities, we compute for each of them the number of sent packets, the number of received packets and the percentage of lost packets. The results are given in table 1. For example, for a packet size s=1024 bytes, quality Q1 information is obtained like this:

- 1Mb/s / (s=1024*8b) * (t=180s) = 23040 sent packets;
- 0.6Mb/s / (s=1024*8b) * (t=60) + 1Mb/s / (s=1024*8b) * (t=120) = 19968 received packets (Q1 is greater than available bandwidth for the first 60 seconds, so only 600kb/s are received from the 1Mb/s sent);
- 23040 – 19968 = 3072, i.e. 13.3% lost packets.

The ideal case appears when the bitrate is adapted to available bandwidth, i.e. 512kb/s for the first minute, 2Mb/s for the second minute and 3Mb/s for the third minute. It leads to no lost packets and maxb*time sent and received packets, where maxb is at any moment the maximum bitrate less than or equal to the bandwidth.

First, this table confirms what we said before: no static bitrate is suitable for the whole transmission. Second, the adaptation is clearly better than each of the static bitrates. The only quality with no lost packets

is Q0, but it is worse than ideal case, since it has only 11520 received packets compared to 42240 for the ideal case. Note that the number of received packets for Q3 is higher than the ideal case, since it leads to lost packets; e.g. for the first 60 seconds, the ideal case sends at 512kb/s and receives the same, while Q3 sends at 3Mb/s and receives 600kb/s.

Finally, it is difficult, or even impossible, for the user or some program to decide at the beginning of a video transmission which static bitrate is the best to use during the whole transmission. On the contrary, the adaptation is automatic, i.e. it does not involve any action from the user.

3. RELATED WORK

3.1. Methods using only the application layer

In this approach only the application layer is modified. The application uses information provided by lower layers. The characteristics of video encoding determine how to adapt the bitrate of the video to the available bandwidth. For example, if the video changes the quality, a flow commutation is done. Our method belongs to this category.

RAAHS, Rate Adaptation Algorithm for HTTP Streaming RAAHS proposes an adaptation algorithm for adaptive video streaming over HTTP [12]. This algorithm, applied on client site, is based on the loading time of a video segment SFT (Segment Fetch Time) to detect congestion and calculate the transmission rate over HTTP. For that, this algorithm compares the SFT with the segment duration MSD (Media Segment Duration) which has a value of 5 to 10 seconds. Basically, if the loading time of an SFT segment is greater than its duration, then the throughput is lower than the bitrate of the video. Otherwise, the available bandwidth is greater than the bitrate.

AHDVS, Akamai HD Video Streaming AHDVS [13] employs HTTP connections to transmit videos between server and client. Videos are encoded at five levels. The client provides regular information to the server, such as the bandwidth estimated by the receiver, the size of the receive buffer, the number of frames received per second and the current bitrate received. To decide the quality to transmit, the algorithm uses the receiver buffer size q and a predefined threshold q_t as follows:

$$r = l\frac{\max((1+\frac{q_t-q}{q_t})100, 10)}{100} \quad (1)$$

where r is the data rate at which the application should supply the TCP buffer and l is the current bitrate. In this equation, when q is equivalent to q_t, r is equivalent to l, and when $q_t - q$ increases the quality increases.

QAC, Quality Adaptation Controller QAC [14] uses feedback control theory for video streaming. The adaptation of the video is done on the server side. The QAC sender chooses the highest bitrate which makes q greater to a predefined threshold q_t, while keeping at the same time the data rate under the available bandwidth. So, the buffer has always data to transmit.

CBVA, Content-Based Video Adaptation CBVA is an adaptation method for streaming of stored video files [15]. CBVA encodes each video in multiples qualities (different bitrates and frame rates) and for each one it creates a group of operating points (combination of frame rate and frame quality). When available bandwidth changes, CBVA changes the streamed video to another operating point. There is a deadline for each operating point. The video quality is increased if this deadline is highly respected, otherwise it is decreased.

Positioning The methods described in this category are divided into two types: methods performing adaptation at the sender side, and on the receiver side. Although the latter can take into account the size of the receive buffer, they require overloading the network with additional traffic between client and server to make the adaptation. In addition, they require a change on the client applications, which is not practical.

For the first type, we note that either the above methods are optimised for a particular type of transport protocols such as TCP, or they operate under certain conditions. This makes these solutions less general, applicable only in certain cases.

3.2. Methods using several layers

Another approach is that *both layers change*: the lower layers give feed-back on the network conditions to application layer, which acts accordingly (for example, it adapts the video bitrate).

VTP, Video Transport Protocol Video Transport Protocol (VTP) [16] was specifically designed to transmit videos encoded in MPEG-4. VTP controls the transmission every k packets. The transmitter sends a request for acknowledgements and receives replies. This exchange allows it to calculate the RTT, the sending rate, and subsequently the best bitrate to be used. VTP has the drawbacks inherent to the implementation of congestion control at this level. It also requires a modification of the application on the receiver side, and there is a risk that is not compatible with TCP (TCP-friendly).

The method described in [8] acts the same way as VTP, but it creates a new transport protocol.

Buffer-driven The Buffer-driven adaptation method, described in [17], uses the occupancy of the receiver buffer to infer which video quality should be chosen and streamed. The occupancy of the receiver buffer is

done at the sender by the following equation (presented in [18]):

$$B_E = B_C + \left(\frac{pktnum * s}{i} - R\right) * RTT \qquad (2)$$

where *pktnum* is the number of packets, s the packet size, i RTCP interval, R the encoding rate and B_C the last buffer occupancy. The method simply tries to maintain B_E inside two thresholds.

Positioning The methods proposed in this category all share the same idea: to transmit streaming video in an adaptive manner, several layers are changed. These methods require significant changes to the operating system, which make them difficult to deploy.

3.3. Methods proposed by major companies

The three commercial products IIS Smooth Streaming [19], Adobe HTTP Dynamic Streaming [20] and HTTP Adaptive Live Streaming [21] are at high-level similar. A manifest file containing detailed information about how video is cut is available on the server. Clients get it, parse it and regularly choose the part of the video to demand: what quality, from which time to which time, what resolution and so on. On the other hand, the format of the manifest file and client availability is different for each of the three solutions.

MPEG DASH MPEG DASH (Dynamic Adaptive Streaming over HTTP) [11] is a recent ISO standard technically similar to the three HTTP-based methods above, but vendor-neutral. It is not bound to any codec. The adaptation is made solely by receiver.

Positioning These methods use HTTP over TCP. They have several practical advantages, the most proeminent being the use of already established protocols which pass firewalls. Some drawbacks are:

- As they use TCP, they have the drawbacks inherent to TCP for hard real-time video streaming, such as videoconferencing: 100% reliability through retransmission, which blocks the receiver until the next packet has been received (this is more exacerbated when the same packet is lost twice by the network: its initial transmission and its retransmission for example). Moreover, TFRC is a better choice for video streaming [3], but it cannot be used with HTTP.
- All the clients need to be modified (e.g. by installing a plug-in).
- It cannot be used in conjunction with other optimisation methods (packet prioritisation, selective retransmission etc.), since they are bound to HTTP and TCP standards.

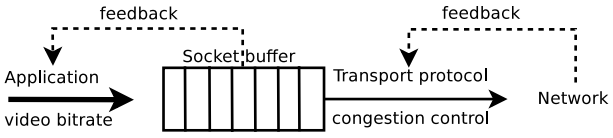


FIGURE 2. Video data flow on the sender.

4. VAAL, VIDEO ADAPTATION ALGORITHM

To be able to know whether to increase, maintain or decrease the video quality, the sender application should have some information about the available bandwidth. In our proposed method VAAL, the bandwidth availability is measured through the number of packets which failed to be written to the transport protocol socket (buffer overflow). The higher the number, the less the available bandwidth. Regularly, VAAL computes this number of packets and changes the video bitrate accordingly. VAAL requires a transport protocol with a network congestion control, no matter which one. It uses a simple algorithm (see appendix Appendix A). Finally, it is made at application layer and does not need any change to system kernel.

Note that while doing adaptation, the sender should try to minimise the quality fluctuation. It is not preferable, in our opinion, to change quality at each sent packet but on a longer scale (each 100ms, each 2sec or each I image for MPEG video for example). However, some codecs may have constraints on the time when the quality changes.

4.1. VAAL explanation

As shown in figure 2, the server application writes packets to the transport protocol socket buffer at a rate equal to the current video bitrate. The transport protocol has a congestion control which gives the rate at which the packets leave the socket buffer (and afterwards the machine, and enter the network). VAAL goal is to adjust the video bitrate to the rate estimated by the transport protocol. Thus, VAAL algorithm is divided in two steps: *discovery of the network conditions* (available bandwidth based on buffer overflow) and *quality selection* (the action to be done). These two steps are executed for each period of n fixed seconds.

The discovery of network conditions works as following. When the application generates packets at a higher rate than the transport protocol can send to lower layers (network), packets are buffered. If the buffer becomes full, the new packets generated by application will fail to be written (buffer overflow). Thus, VAAL monitors the available network bandwidth through the transport protocol socket buffer overflows when the application tries to write a packet in it. During each period, VAAL computes the percentage of these failed packets, which we will call *write failure*

percentage (WFP). Consequently, WFP is an indication of the network conditions: the bigger the WFP, the less the available bandwidth.

The quality selection (adaptation) works as following. At the end of each period, VAAL reads the WFP (given by the first step) and acts like this:

- If WFP is null (no packets failed when written to buffer) VAAL chooses the next higher quality level (higher bitrate) unless the quality is already the highest.
- Elsewhere, if WFP is tolerable (smaller than 5%), the quality is maintained at the same level. ITU.T G.1070 [22] recommends that the end-to-end IP packet loss rate in video streaming should be less than 10%. Hence, we chose a threshold of 5% of packet loss rate at the sender buffer (WFP $<$ 5%), the other 5% being left to handle the network losses.
- Finally, when WFP is greater than 5% and unless the lowest quality is already in use, VAAL searches for the highest quality q' which fulfils:

$$q' \leq q(1 - \text{WFP})r \tag{3}$$

where q is the current quality. In this formula, $q(1-\text{WFP})$ represents the bandwidth available for the period which has just ended, while $r > 0$ is a parameter (*aggressiveness*) which allows us to choose a quality with a bitrate different than the available bandwidth.
Normally, the next quality q' should be smaller than q, but depending on the aggressiveness r, the quality could remain the same or even increase.

As mentioned before, VAAL requires a transport protocol with a network congestion control, no matter which one. Also, VAAL is especially useful in video conferencing (video on-the-fly) because there is no need to re-encode the video, just changing the encoding rate.

4.2. Implementation

We have implemented VAAL video adaptation at the application layer on a GNU/Linux machine with 2.6.35 kernel (without any change to system kernel). The period of time n is 2 seconds. Packets which failed to be written in the socket buffer are deleted (i.e. they are not stored so that they can be retransmitted later), in order to reduce delay. The program uses DCCP as transport protocol together with TFRC as congestion control (DCCP was implemented in the kernel a few years ago).

This implementation represents an enhanced version of our previous works [23] and [24]. In the version of VAAL presented in this paper we put them together and studied four enhancements through new deeper experiments. The first three show better performance while the last one gives similar results.

4.2.1. Initial bitrate

Starting the connection with a high bitrate will cause transport protocol buffer to drop a high number of generated packets, which has a negative impact on video quality at the beginning of each video. "Slow-start" solutions for the bitrate at the beginning of a video streaming have already been proposed, but they are not generally applicable, because the adaptation can often be done only at coarse grain (e.g. each 2 sec.) So in VAAL we took a simple and general solution: we decided not to start the transfer with a high bitrate but with a small one. Our choice was also not to use the smallest bitrate because it will take a few seconds to reach the highest bitrate if the bandwidth is high enough to support it. In our current implementation of VAAL, the initial video bitrate used is 1Mb/s.

4.2.2. WFP computation interval

WFP is the parameter used by VAAL to estimate the available bandwidth. We have considered two ideas for the interval of time on which WFP is calculated: use either the last RTT (Rount Trip Time), or the interval of time for the last p sent packets; p should not be smaller than 20, in order to allow a percentage of failed packets (WFP) less than or equal to 5% to appear, cf. the quality selection above. However, because the RTT may be very small (depending on network topology) and the number of packets during this period could be very small too, we choose instead a hybrid solution using both ideas, where the interval of time is max (last RTT, last 20 sent packets).

4.2.3. r aggressiveness value

r represents the aggressiveness of the transfer. As previously shown (equation 3), the next quality chosen q' is the highest available quality which verifies $q' \leq wr$, where w is the bandwidth. If $r = 1$, the next quality will be at most w, while if $r = 1.1$ for example, the next quality could be as great as $1.1w$. However, in this latter case it does not necessarily mean that the quality will always be greater than w; if for example available qualities of the transmitted video are increasing as powers of 2 (512kb/s, 1Mb/s, 2Mb/s etc.), then the highest available quality smaller than or equal to $1.1w$ is somewhere between $1.1w/2$ and $1.1w$, with an average of $1.1w*3/4 = 0.825w$, which is smaller than w. On the other hand, if all qualities are available (i.e. supported by a codec during an on-the-fly encoding), then the next quality will always be near or equal to $1.1w$, which is greater than w, hence not a good solution.

As an numerical example on the influence of r to the chosen quality, let's suppose that the available bitrates are the ones above (512kb/s, 1Mb/s, 2Mb/s etc.), r is 1.1, and that in some point of time the current bitrate is 1Mb/s and the measured WFP is 7%. According to equation 3 in previous section, $q' \leq 1\text{Mb/s} * (1 - 0.07) * 1.1$, that is $q' \leq 1.023\text{Mb/s}$. This means that the

quality of the flow remains at 1Mb/s in that point of time. However, if WFP was 10%, then $q' \leq 0.99$Mb/s and the quality would decrease to 512kb/s.

As a conclusion, r should be chosen carefully and be based on the available qualities of the video: the higher the bitrate ratio between consecutive qualities, the higher the r value. In the current version several values for r have been tested and $r = 1.1$ is used.

4.2.4. Real vs theoretic bitrate

It is known that sometimes encoders cannot encode a video to a specific bitrate, i.e. the bitrate of the video generated is not equal, but (more or less) near the bitrate asked. We denote by q_t the theoretic (or asked) bitrate for a video and by q_r the real (or generated) bitrate.

As previously specified, when VAAL is in quality selection step and after WFP calculation, next quality q' (for the following 2 seconds) is chosen so that $q' \leq q(1 - \text{WFP})r$ (formula 3), where q is the quality used in the past 2 seconds. q and q' can have two different meanings:

- For q' (in the future): The bitrate for the next 2 seconds can sometimes be predicted (for example for video streaming using pre-stored files), sometimes not (for example for interactive video conference, when the codec is not very precise). If prediction is possible, q'_t or q'_r can be chosen, otherwise only q'_t can be chosen.
- For q (in the past): At any moment, the real bitrate q_r used for past 2 seconds is available. However, the theoretical bitrate q_t used 2 seconds ago exists only if in the previous step (q', in the future) it was chosen.

Since q'_r is not available for any kind of video transmission, we prefer to choose q'_t for VAAL. There is now a choice between q_t or q_r; in the current version both values have been tested and they gave similar results, and we decided to *use q_t for next quality decision.*

4.3. Amelioration of our implementation

We noticed that our VAAL method has two possible kinds of drawbacks. The first one is the overestimate of the bandwidth, which leads to a higher transmitted video bitrate (over-bitrate) than available bandwidth. This over-bitrate may happen in the following cases for example: 1) In the middle of the two seconds (period of time used by VAAL to check network conditions), the bandwidth abruptly goes down, so many packets will be lost until the end of this period of two seconds. 2) We over estimate the bitrate when we switch to a higher bitrate, but the bandwidth is smaller than that (VAAL cannot predict the exact bandwidth). In the following section we propose an algorithm to minimise the number of lost packets. It uses a history of bad decisions, allowing to know which bitrate overestimated the bandwidth, in order not to try it for a while.

The second one is the video under-bitrate (video bitrate is smaller than the network bandwidth) due to a underestimation of the bandwidth. In the same manner as above, this can appear: 1) When in the middle of the two seconds, the bandwidth abruptly goes up. 2) When the bitrate is switched to the next higher one, but the bandwidth is even higher than that.

A solution for these drawbacks is to adapt to network conditions more often, but this is subject to codec constraints.

5. ZAAL, ZIGZAG AVOIDANCE ALGORITHM

A video adaptation method such as VAAL presented previously can sometimes lead to a continuous switching between two qualities: one smaller than available bandwidth and the second greater. For example, if a user is connected to Internet via a 2.5Mb/s link and the video is available in 2 and 3Mb/s qualities, then an adaptive streaming algorithm will constantly switch between these two qualities. Obviously, the best solution would be to stay with 2Mb/s much more time before retrying 3Mb/s quality. Since the adaptation algorithm does not know when the available bandwidth could be larger than 3Mb/s, it should retry a superior bitrate from time to time. This constant change induces what we call ZQS *zigzag quality switching* problem (the zigzags are also called oscillations). We noticed this phenomenon in our first experiments (see figure 5(a) for a clear example, where during the first minute the video bitrate is continuously toggling between 0.5 and 1 Mb/s).

If this issue is already known (see [25] for instance), only few papers treat it, and they do not solve it completely. For example, in [26] the receiver uses a back-off timer for each layer of the video. If a level led to lost packets, the receiver goes back to previous level and the timer for the level with losses is multiplicatively increased. Authors of [27] present a way for smoothing sent video bitrate and reducing the frequency of video quality changes. The main idea is that the application does not switch to a higher video quality until the sender is certain that the video will continue playing even after a reduction of the congestion window. A metric to evaluate the jerkiness of a video, given by the number of quality changes, is given in [15]. It uses a formula to calculate an effective frame rate for each video. This formula is used to limit the number of quality changes for each defined window during the transmission.

In this section we introduce a simple, easy to implement and scalable solution, called ZAAL (**Z**igzag **A**voidance **Al**gorithm), to this problem (for detailed information the reader is referred to [24]). We present also the integration of ZAAL in the method of video adaptation at application layer on the server (VAAL).

VAAL behaves as explained in section 4 except that when the quality is to be increased (in the quality selection phase), VAAL consults ZAAL for whether a superior bitrate is allowed or not. If ZAAL decides that the increase does not lead to zigzag, then VAAL increases the quality; otherwise, VAAL maintains the current level.

Note that ZAAL is not an adaptation method. It is used only to prevent an adaptation method from frequently switching the video quality. The only information ZAAL needs comes from the adaptation method, whether some bitrate causes lost packets or not.

5.1. Zigzag-avoiding algorithm overview

ZAAL algorithm works by avoiding constantly using bitrates higher than the available bandwidth. For that, it maintains an successfulness value for each bitrate, called *successfulness* in the following. When the adaptive algorithm considers to increase bitrate (and only in this case), ZAAL checks if the successfulness of the higher bitrate is lower than a threshold, called β; if this is the case, a higher bitrate *cannot* be chosen. Otherwise said, application uses a higher bitrate i only if its successfulness $S_i > \beta$. After this process, the successfulness is updated.

More precisely, ZAAL algorithm uses the *successfulness* value each time an adaptation period ends (e.g. each 2 sec. in case of VAAL). Successfulness value is calculated separately for each bitrate (with different weights), denoted by S_i, which indicates if bitrate of index i can be used for the next period or not. As such, this value expresses application last attempts to use the corresponding bitrate. In brief, when a bitrate generated failed packets to the transport protocol buffer, corresponding successfulness value is greatly reduced; when a bitrate was successful, the successfulness value is greatly increased; finally, when a bitrate has been used for a long time, the successfulness value corresponding to the higher bitrate is slowly increased. As a general rule, the smaller the successfulness value, the more the corresponding bitrate caused failed packets and application must avoid using it.

The successfulness S_i (where i is bitrate index) of each bitrate, which changes according to the history of that bitrate, is calculated using an EWMA algorithm (Exponential Weighted Moving Average). Using EWMA allows to give greater weight to recent history compared to older history, since obviously current bandwidth is better expressed by recent bitrate usage than by older ones. Additionally, different weights are used, based on the bitrate involved.

At the beginning of a video transmission, all S_i values are set to 1. Then they are calculated each time the application wants to adapt the video bitrate to the available bandwidth, using the following general formula:

$$S_i = (1 - \alpha/d)S_i + s(\alpha/d) \quad (4)$$

where:

- s is the successfulness at the time of measurement (the current "observation") and can be either 0 or 1. 0 value is used when the bitrate did not give good results (hence its successfulness value will decrease). 1 value is used when the bitrate gave good results, either because the bitrate did not cause failed written packets, or because the bitrate was not used recently (hence its successfulness value will increase).
- α is the degree of weighting increase/decrease, a constant smoothing factor between 0 and 1. A higher α discounts older successfulness values faster.
- d is a division factor allowing to speed up or slow down the value increasing depending on the bitrate involved. In our algorithm, d has three values: 1, 2 and 4. For $s = 1$, the greater the d, the slower the increasing of the S_i value. The value of d depends whether the application increases, maintains or reduces bitrate.

In our experiments we intuitively set $\alpha = 0.3$ and $\beta = 1 - \alpha = 0.7$. Other values were analysed but either they lead to high number of adaptation iterations for the algorithm to converge, or they minimise its effects.

6. EXPERIMENTAL RESULTS

Figure 3 shows the real network used to realise experiments with the program presented in the previous section. A video streaming connection is made between a sender and a receiver with wired interface for both. An intermediate machine (called *shaping machine* in the following) is added between the sender and the receiver[2]. This shaping machine has two interface cards: a wired interface connected to the sender and a wireless one (54Mb/s) connected to an access point (AP). The receiver is connected to the same AP via its wired interface. We chose a wireless network in order to be nearer the reality, since wireless networks are commonly used today to access Internet and such networks are more challenging for the algorithms involved in video adaptation. The video streaming uses a real video, available in four qualities 3Mb/s, 2Mb/s, 1Mb/s and 512kb/s, as mentioned before. The video has 180s and the real bitrate during each second for each of the four asked bitrates is given in figure 4.

Three series of tests are done. For the first series (traffic shaping series), just one flow is present at any

[2]We preferred to add an intermediate machine since making the traffic shaping on DCCP sender is not working currently (see thread "DCCP_BUG called" on DCCP mailing list, available at `http://www.spinics.net/lists/dccp/msg04264.html`), making it on the access point is not interesting either, because it has an older linux kernel, and making it on the receiver machine does not give accurate results.

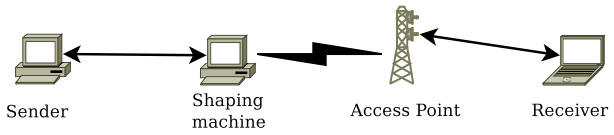


FIGURE 3. Network topology used for experiments.

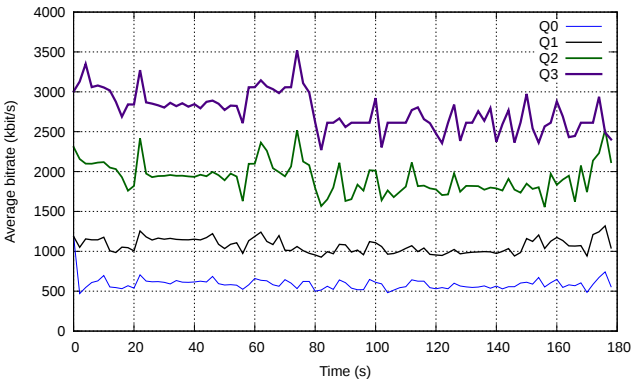


FIGURE 4. Real bitrate of the four theoretical (asked) bitrates 512kb/s, 1Mb/s, 2Mb/s and 3Mb/s.

moment during the video transmission, which can thus use all the available bandwidth. On the other hand, the output bandwidth of the shaping machine is changed three times to simulate a variable bandwidth: 600kb/s for the first minute, 2300kb/s for the second minute, and traffic shaping was stopped for the last minute (hence the output bandwidth is the original wireless bandwidth) (see figure 1). For the second series, two numbers of concurrent flows are present during the whole transmission: five and ten flows (called in the following *flows without gap* series). In fact, five flows at maximum bitrate are comparable to the bandwidth of the network (5*3Mb/s = 15Mb/s, which is about the bandwidth provided by a classical 54Mb/s wireless network), while 10 flows clearly exceed the bandwidth. This allows to see what happens when multiples flows sense the available bandwidth, especially to check if this leads to a wide oscillation in performance. In fact, more video adaptation frequencies will be seen as the number of flows increases from five to ten. In the third series, a various number up to twelve flows are present at the same time. Each flow starts 10 seconds after the beginning of the previous flow, except the first one which starts at time 0 (called in the following *up to twelve flows with gap* series); this is presented more in detail in section 6.2.2.

For the second and the third series, each flow waits before starting for a random number between 0 and 2 seconds, to prevent that each two seconds all flows change the bitrate at the same time. During experiment execution, all the flows use the same algorithm (i.e. all of them use adaptation, or all of them use some fixed quality).

The series above consist in executing the same

Method	*First minute*	*Second minute*	*Total*
Without ZAAL	13	10	23
With ZAAL	3	1	4

TABLE 2. Number of zigzags with and without ZAAL.

experiments several times. First series has ten repetitions, and second and third series have five repetitions. Depending on the network conditions one method can take advantage over the others just because during its experiment the network conditions were good; with five or ten repetitions, the chance for something to happen again and again is very small. Only one representative experiment is presented here, and the overall average of all experiment repetitions. Note that there is no retransmission for lost packets in all our tests (as given by DCCP).

In the remaining of the section, we first confirm that ZAAL is a useful addition to VAAL, and afterwards evaluate VAAL+ZAAL.

6.1. ZAAL usefulness

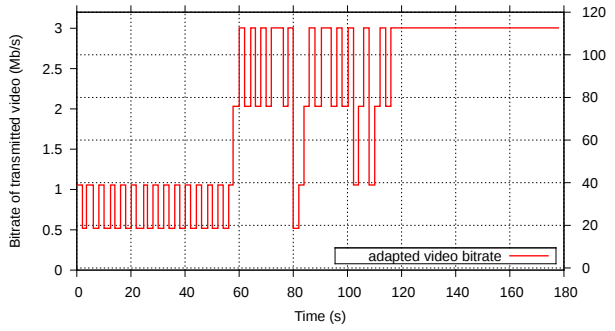
6.1.1. Zigzag avoidance of ZAAL

The experiment we present uses one flow in case of traffic shaping. Figure 5 presents the results. The x-axis represents the time from 0 to 180s, the duration of a video transmission, and the y-axis shows the video bitrate (VAAL with or without ZAAL). As expected, ZAAL minimises the zigzag effect: during the first minute in figure 5(a) (where ZAAL is not used), the video bitrate is continuously toggling between 0.5 and 1Mb/s, while when ZAAL is used, in figure 5(b), application uses 0.5Mb/s bitrate most of the time. The same conclusions can be drawn from the second minute. During the last minute, bandwidth is wide enough to support 3Mb/s, and ZAAL finally allows this bitrate.

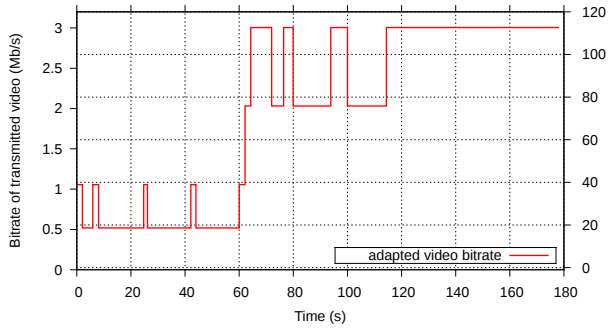
More specifically, table 2 presents the number of zigzags during the first and the second minute (there is no difference during the third minute). It is clear that ZAAL leads to much fewer zigzags (3 against 13 in first minute, 1 against 10 in second minute, so about 85% less in total). Naturally, this has a very big impact on the video quality perceived.

6.1.2. Impact of ZAAL on transmission performance

Even if reducing packet loss rate is not the main goal of ZAAL, we investigate how ZAAL affects it. We consider the number of received packets and the number of lost packets. Video quality over network transmission is more affected by packet losses rather than video bitrate value. Table 3 presents numerical results of the same experiments. For one flow in case of traffic shaping, it is clear that the number of received packets is lower when ZAAL is used (e.g. only 39865 received packets for ZAAL compared to 42043 without ZAAL), because of ZAAL preventing higher bitrate for some period. On the other hand, when ZAAL is used the flow has 30%



(a) without ZAAL: many zigzags occur



(b) with ZAAL: few zigzags occur

FIGURE 5. Quality adaptation for one flow in case of traffic shaping (using the *same* traffic shaping).

fewer lost packets (under the same network conditions). The average of the ten concurrent flows gives yet better results for ZAAL, i.e. number of received packets are nearly equal but loss rate is about 50% smaller.

To summarise:

- in the first experiment, it cannot be easily decided which is better in terms of number of sent and received packets, but using ZAAL is more useful because it avoids the zigzag and gives a more stable video quality;
- in the second experiment, ZAAL is better in terms of sent and received packets, avoiding the zigzag in the same time.

We can conclude that using ZAAL is better and even if sometimes the number of received packets is lower, it reduces the rate of lost packets while maximising the use of the bandwidth.

6.2. VAAL + ZAAL evaluation

This section presents three results: the bitrate adaptation takes well into account the DCCP buffer feedback (quality variation), our method outperforms the widely-used static encoding (adaptation performance), and several applications on the same server machine are fair when they send packets. For simplicity, we will call ZVAAL the combination of VAAL and ZAAL.

6.2.1. *Quality variation*

In this section we discuss the quality variation made by ZVAAL. As seen before, every two seconds ZVAAL looks at the write failure percentage (WFP) of DCCP buffer to decide if it has to increase, maintain or decrease its bitrate. We present here results for one flow in traffic shaping, and five and ten flows without gap. For better visualisation, we show write *success* percentage, which is simply 1 – write failure percentage (i.e. 1 –WFP).

In the following figures, the x-axis represents the time from 0 to 180s, the duration of a video transmission. The bitrate of transmitted video and the write success percentage are put on the y-axis. Three curves are plotted in each graphic: the current theoretical (asked) video bitrate q_t asked by ZVAAL during transmission and used to calculate next quality during quality selection step, the real video bitrate q_r at which the sent video is encoded, and the write success percentage.

One flow in case of traffic shaping We present first the video adaptation in case of traffic shaping. The result of this test is shown in figure 6.

As expected, during the first minute, where the bandwidth is limited to 600kb/s, ZVAAL keeps the bitrate at the lowest quality (512kb/s) while trying to sense a higher bitrate (1Mb/s) from time to time, e.g. at seconds 24 and 42. During the second minute, where the bandwidth is limited to 2.3Mb/s, ZVAAL discovers the new available bandwidth and uses most of the time a video bitrate between 2 and 3Mb/s (higher quality). In the last period the shaper is switched off, and ZVAAL sends the highest bitrate (highest quality). The last minute shows also that when bandwidth is higher than the highest quality, switching among qualities does not occur, hence ZVAAL is comparable to static transmission of the highest quality.

Further analysis of this figure confirms the useful properties of zigzag avoidance ZAAL algorithm. First, application waits for at least 2 periods of time before trying a bitrate which has recently caused losses (e.g. at the beginning, 1Mb/s did not work between 0s and 2s, hence it was retried not at 4s, but at 6s). Second, when a bitrate causes losses for many consecutive times, application waits more and more time to retry it (e.g. 1Mb/s at 0s, then at 6s, afterwards at 24s, and finally at 42s). Third, the maximum period during which the video quality was prevented to increase is 16s. (e.g. the first unsuccessful attempt finishes at 26s followed by the next successful attempt at 42s); during that time there were no losses (bandwidth much higher than bitrate), however ZAAL correctly prevented the bitrate increasing.

Five flows without gap In this test five flows are running and using the available bandwidth. Figure 7 shows the results for one flow of this test. Quality

	Traffic shaping			*10 concurrent flows*		
Method	*Sent pkts*	*Rcv pkts*	*Lost pkts*	*Sent pkts*	*Rcv pkts*	*Lost pkts*
Without ZAAL	47795	42043	5752 (12%)	41191	32307	8884 (21%)
With ZAAL	43548	39865	3683 (8%)	36713	32477	4236 (11%)

TABLE 3. Number of sent and received packets (average of all flows) with and without ZAAL.

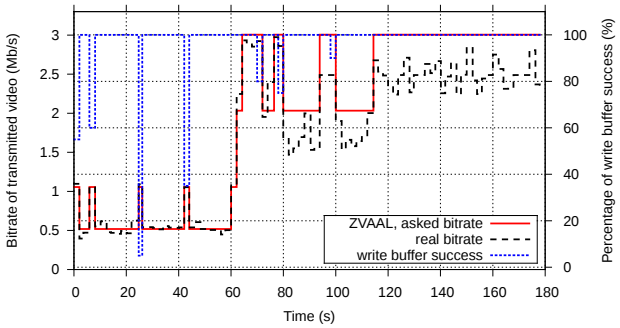


FIGURE 6. Quality adaptation for one flow in case of traffic shaping.

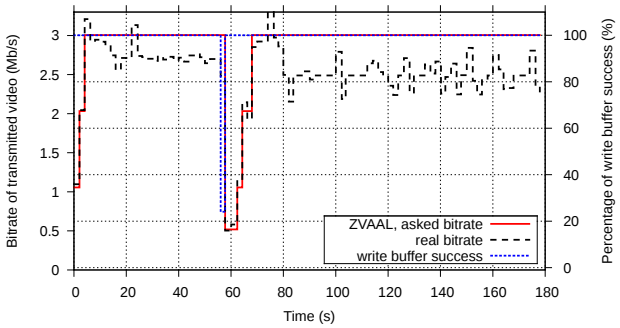


FIGURE 7. Quality variation for five flows without gap (one representative flow).

selection starts the connexion with video bitrate of 1Mb/s, as explained before, and then increases slowly to reach the highest bitrate quality. It can be noticed that the quality is most of the time at 3Mb/s, because available bandwidth is sufficient to let five flows send at high quality. When the available bandwidth is not sufficient anymore, for example at second 58 due to some interference on the wireless link (write success rate is very low), the quality is directly switched to the smallest. On the contrary, when the write success rate is again 100%, the quality is increased slowly (switched to the available higher bitrate every two seconds, for example at seconds 62, 64 and then 68.

Figure 8 presents the bandwidth used by each flow, together with the total bandwidth for all the 5 flows, for one repetition. The bandwidth is given by the real bitrate, not the asked bitrate, hence the changes in bitrate appearing in the figure are generally not quality switching done by the video adaptation, but the varying real bitrate generated by the video encoder itself. Each flow in this figure gives similar conclusions as the representative flow. It can also be seen that

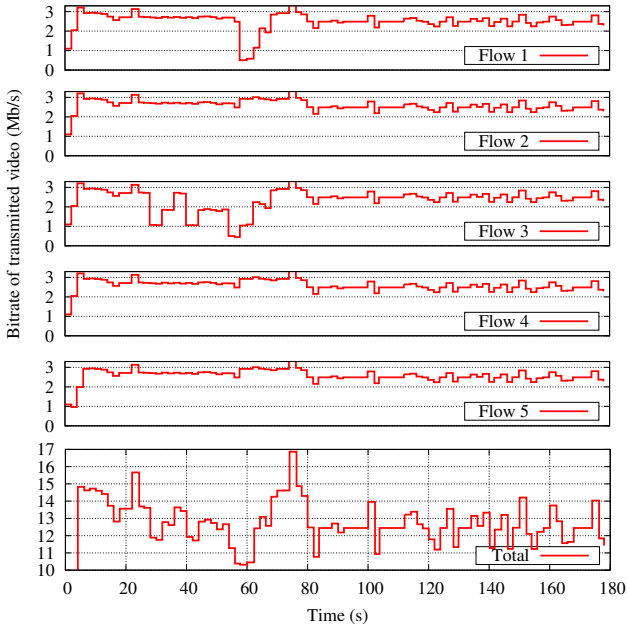


FIGURE 8. Quality variation for five flows without gap (each of the five flows and their total).

the average total bandwidth is between 12 and 13 Mb/s, i.e. about 2.5Mb/s per flow, which confirms that flows generally use the highest available bitrate, which cf. figure 4 is around 2.7Mb/s.

Ten flows without gap This test is similar to the previous one but for ten flows. Figure 9 presents the results for one flow of this test. Like for five flows, it can be noticed that when the buffer success rate is very low the quality chosen by ZVAAL is very low, e.g. at 46s. Also, given that there are ten flows which compete for the bandwidth, quality switching can be noticed too; in this way, ZVAAL insures that the quality is a function of the bandwidth. The average available bandwidth is somewhere between 1Mb/s and 2Mb/s, so ZVAAL is most of the time choosing these bitrates.

Figure 10 presents the bandwidth (corresponding to the real bitrate, not the asked bitrate) used by each flow, together with the total bandwidth for all the 10 flows, for one repetition. The same conclusions as the representative flow can be drawn for each flow. It can also be seen that the average total bandwidth is about 16 Mb/s, i.e. around 1.6Mb/s per flow, and that all flows choose most of the time the qualities near the 1.6Mb/s average, i.e. 1 and 2 Mb/s.

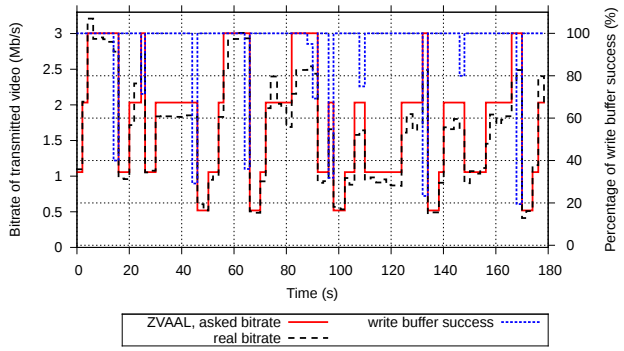


FIGURE 9. Quality variation for ten flows without gap (one representative flow).

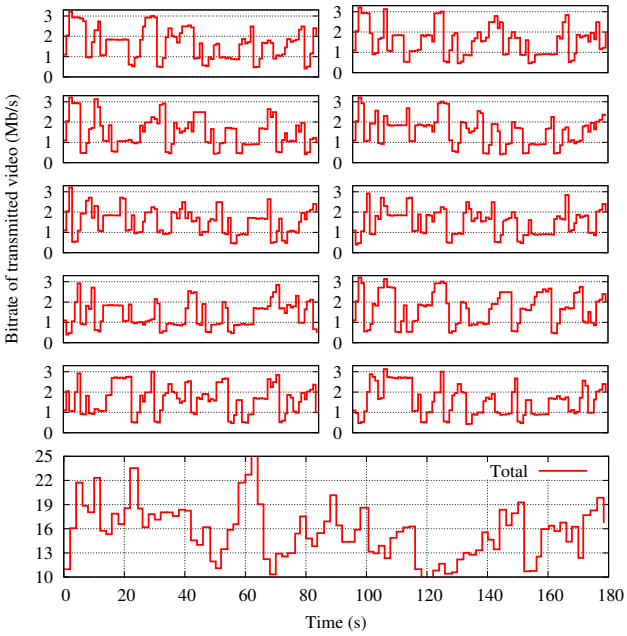


FIGURE 10. Quality variation for ten flows without gap (each of the ten flows and their total).

6.2.2. Adaptation performance

In order to find out if adaptation is useful, we compare ZVAAL with the same application (using DCCP) but without adaptation. We recall that our method optimises the network part of a video transmission, and can be used in combination with other video optimisation methods. As a consequence, the metrics used are the number of received packets and the number of lost packets. We assume that if a new method is able to maximise the number of received packets while minimising the number of lost packets, it will ameliorate the received video quality.

The results for ZVAAL are taken from the tests done before. For DCCP without video adaptation, the same three series of experiments have been done, separately for each of the four qualities (Q3=3Mb/s, Q2=2Mb/s, Q1=1Mb/s and Q0=512kb/s).

In each graphic below there are ten curves, which

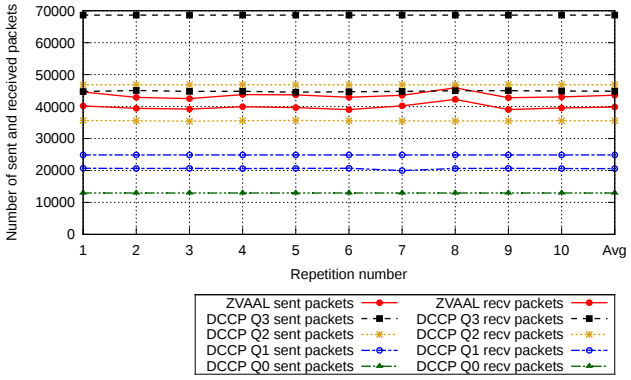


FIGURE 11. Comparison of number of sent and received packets for one flow with traffic shaping.

give the number of sent and received packets for each streaming quality: Q0, Q1, Q2, Q3 and ZVAAL (in order to distinguish them more easily, the curves for sent and received packets for each type use the same point style). Note that, even if all the curves are put on the same graph, the execution is done at *different* times. Also, even if the curves use lines for better visualisation, the flows and repetitions (on x-axis) are independent.

One flow in case of traffic shaping The results are shown in figure 11 for the 10 repetitions of the test. As a first note, it shows that the results for the repetitions are very close to each other (similar number of sent and received packets), which was expected since there is only one flow involved in each test. The comparison between the methods gives:

- ZVAAL number of received packets is smaller than DCCP Q3, but there is a huge difference between them regarding the loss rate (about 35% of lost packets for DCCP Q3 versus only 8% for ZVAAL). So ZVAAL is better than DCCP Q3.
- ZVAAL number of received packets is 11% greater than DCCP Q2, while at the same time its number of sent packets is 7% smaller. Loss rate for DCCP Q2 is very high, about 25%. So ZVAAL is clearly better than DCCP Q2.
- It is clear that ZVAAL is better than DCCP Q1 and DCCP Q0.

As a conclusion, ZVAAL outperforms any static method in case of important variable bandwidth.

Five flows without gap Figure 12 presents the results of one representative repetition for the series of experiments with five flows without gap. As expected, the available bandwidth lets pass the five flows with the highest bitrate. The number of sent and received packets for ZVAAL is comparable to classical transmission with Q3 quality. It can also be noticed that the average number of received packets for ZVAAL and DCCP Q3 is much higher than for Q2, Q1 and Q0,

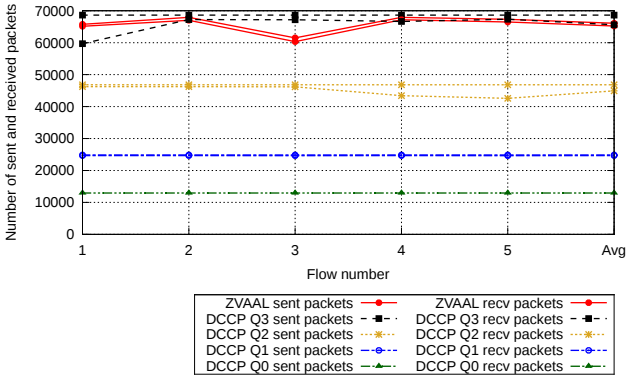


FIGURE 12. Comparison of number of sent and received packets for five flows without gap (one representative repetition).

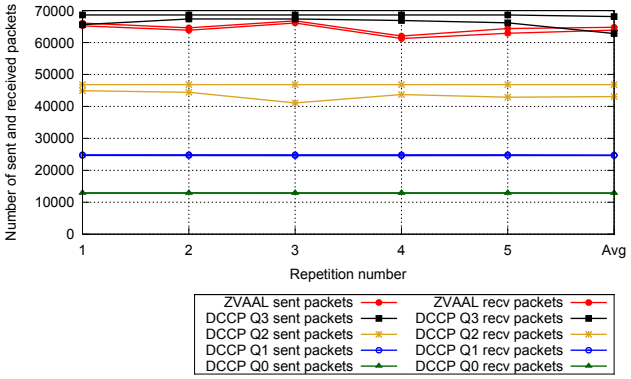


FIGURE 13. Comparison of number of sent and received packets for five concurrent flows without gap (average of five repetitions).

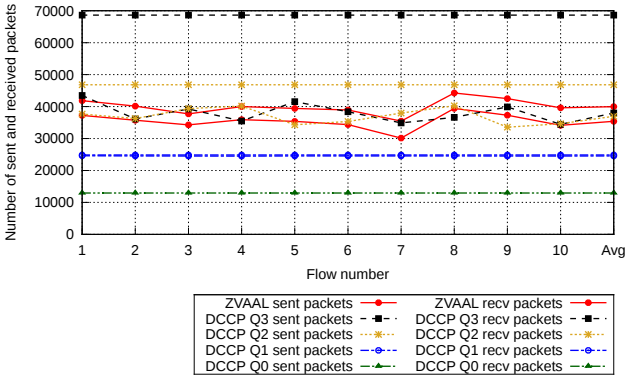


FIGURE 14. Comparison of number of sent and received packets for ten flows without gap (one representative repetition).

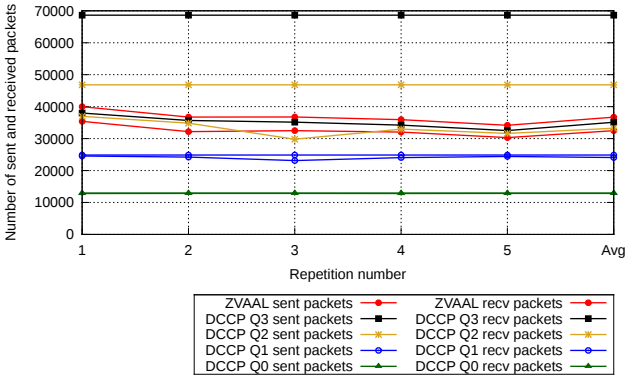


FIGURE 15. Comparison of number of sent and received packets for ten concurrent flows without gap (average of five repetitions).

as expected when the bandwidth is sufficiently high.

Figure 13 (average of the five repetitions) gives the same results. In average ZVAAL is better than the classical transmission because it has a nearly identical number of received packets but with smaller number of sent packets, i.e. loss rate is smaller (about 17% less) when using ZVAAL. Unlike ZVAAL, when classical transmission meets some difficult network conditions, e.g. repetition 5, it does not decrease the number of sent packets. So it is not able to adapt to the network conditions.

As a conclusion, even if available bandwidth is high enough for all concurrent flows at maximum quality, using ZVAAL provide always better performance by keeping loss rate very low.

Ten flows without gap This series is similar to the previous one but for ten flows. Figure 14 presents one representative repetition. It can be seen that all flows using ZVAAL adapt their sending rate to the available bandwidth, i.e. the difference between sent and received packets is similar for all the flows. Moreover, Q2, Q3 and ZVAAL flows have 4 to 5 percent of received packets more than ZVAAL for this representation test. On the other hand, unlike classical transmission, all ZVAAL flows loss 46 to 85 percent less than Q2 and Q3 flows by reducing their number of sent packets to adapt to the network conditions.

Figure 15 (average of all the five repetitions) shows that the number of received packets for all flows is much lower compared to the previous test (so do DCCP with Q3 and Q2), which was expected given the increased number of flows (5 to 10) sharing the same link. The five repetitions shows again similar number of received packets for Q3, Q2 and ZVAAL, but much fewer packet losses for ZVAAL (thanks to its adaptation algorithm). For example, DCCP Q3 has a huge number of dropped packets for all its flows, about 49% of its sent packets, indicating that Q3 is indeed too aggressive for this network. This high rate of dropped packets in DCCP buffer affects considerably the video quality at the receiver side. Compared to DCCP Q2, ZVAAL sends 22% fewer packets for a similar number of received packets.

Finally, we can notice that with classical static transmission the application wrongly continues to send

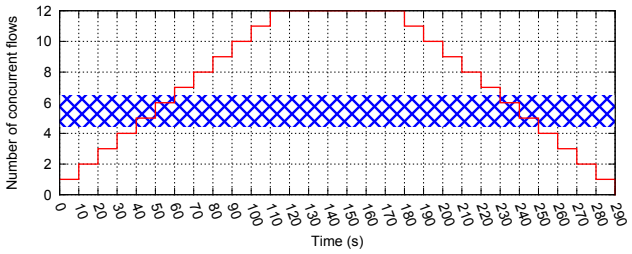


FIGURE 16. Number of concurrent flows at any moment for twelve flows with gap of 10 sec.

at the same rate regardless the network conditions.

Up to twelve flows with gap In this series of tests, up to 12 concurrent flows compete for the network bandwidth. This allows to compare the performance in a dynamic and more realistic situation where the number of flows varies over time. As a consequence, the best static bitrate cannot be known in advance, at the beginning of the transmission. This series of tests also mimics some characteristics of short lived requests/answers of the current Web, for example in these tests flows appear one after the other at the beginning, and disappear one after the other at the end of each test.

As already specified, there are about 10 seconds of gap between the beginning of each two consecutive flows. Figure 16 presents the starting time of the 12 competing flows and the number of concurrent flows at any time; for example flow number 7 starts at second 60, and there are 6 concurrent flows between seconds 50 and 60. Adding 180s to the start time gives the end transmission time for each flow, e.g. end time of flow 7 is 60+180=240s. There is also a blurred zone to indicate theoretically how many concurrent flows at highest quality (3Mb/s) the bandwidth can accept (4.5–6 flows). It can be noticed that the flows with the highest degree of concurrency are 6 and 7: during their lifespan the number of concurrent flows is between 6 and 12. The degree of concurrency experienced by a flow decreases as the flow number is further from 6 or 7 (flows 5 and 8 experience at least 5 concurrent flows, flows 4 and 9 at least 4 concurrent flows etc.) To conclude, as a simple rule, the less the distance between flow number and the middle (6.5), the higher the degree of concurrency.

Figure 17 presents one representative repetition. As expected, flows in the middle, e.g. flows 5 to 8, have the fewest received packets. Additionally, for ZVAAL the curve for the number of received *and* sent packets generally decreases from 1 to the middle (6.5) and increases afterwards, i.e. the flows generally send/receive packets based on the degree of concurrency they experience. This is contrary to static transmission (Q3 and Q2, without adaptation), where the flows in the middle experience a much higher dropped packets rate than others.

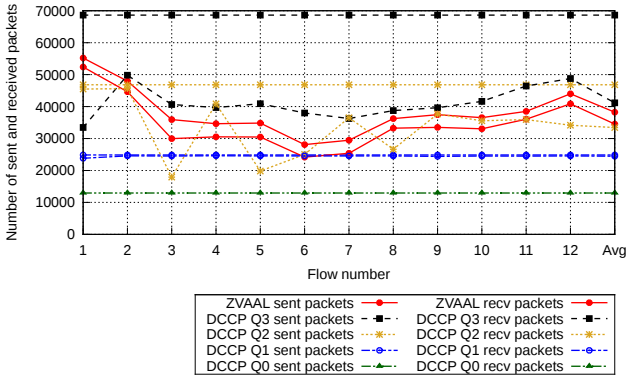
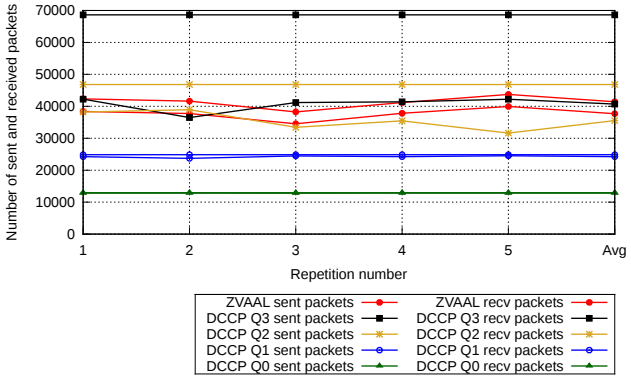


FIGURE 17. Comparison of number of sent and received packets for twelve flows with gap of 10 seconds (one representative repetition).



Average column in detail:

	ZVAAL	Q3	Q2	Q1	Q0
Sent	41418	68623	46818	24858	12938
Received	37683	40698	35531	24232	12828
Lost	3735	27925	11287	626	110

FIGURE 18. Comparison of number of sent and received packets for twelve concurrent flows with gap of 10 seconds (average of five repetitions).

Overall average for all repetitions, shown in Fig. 18, confirms that ZVAAL gives similar results for all repetitions. The last column, Avg, is detailed in the table of the same figure. It shows the superiority of ZVAAL on all classical static transmissions:

- compared to DCCP Q3, ZVAAL has 7% fewer received packets ($40698/37683 - 1$), but 40% fewer sent packets ($1 - 41418/68623$), leading to 87% fewer packet losses ($1 - 3735/27925$);
- compared to DCCP Q2, ZVAAL receives 6% more packets ($1 - 35531/37683$) and sends 12% fewer packets ($1 - 41418/46818$), hence 33% fewer packet losses ($3735/11287$);
- compared to DCCP Q1 or Q0 it is much better, in terms of sent, received and lost packets.

As a conclusion, once again, with video adaptation (ZVAAL) the bandwidth is more efficiently used, especially when it changes dynamically or when the

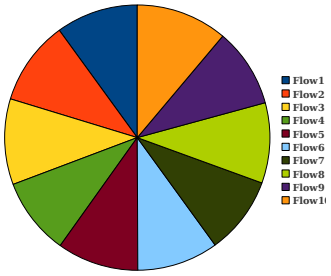


FIGURE 19. Percentage of sent packets by each flow at the application layer using ZVAAL: the percentages are nearly equal.

network experiences some bad conditions.

6.2.3. ZVAAL fairness

We are interested to check if several applications on the same machine are fair with respect to the number of sent packets. The test with 10 flows is used as example, more precisely the test shown in figure 14. Figure 19 shows graphically the percentage of sent packets by each flow at application layer when using ZVAAL. More quantitatively, Jain's fairness index [28], defined as:

$$\left(\sum_i x_i\right)^2 / \left(n \sum_i x_i^2\right)$$

where n is the number of flows and x_i the number of sent packets of flow i, is equal to 0.9975, very close to the ideal value of 1. This result shows that ZVAAL maintains the fairness among concurrent flows on server (i.e. all flows have nearly equal percentage of sent packets).

The fairness presented here measures the fairness induced by our video adaptation algorithm, which we could call adaptation fairness. The absolute fairness depends on this adaptation fairness, and also on the fairness on the network level. The latter one is guaranteed in our method by using a TCP-friendly congestion control. This section has shown that, given a fair (TCP-friendly) protocol in the network, the video adaptation fairness does not by itself unbalance this fairness.

7. CONCLUSIONS

This paper has presented a simple but powerful method (VAAL) to adapt the content of video during streaming, using the buffer overflow method on the server at the application layer. ZAAL, a companion algorithm to reduce the number of oscillations in video quality, was presented too.

Intuitively, this method should lead to much better video streaming performance. Numerous real experiments confirm this hypothesis, i.e. the bitrate used during the adaptation is shaped by the available network bandwidth, and it generally either leads to much fewer packet losses, or avoids a under-utilisation of the network bandwidth. Moreover, the use of a transport protocol (DCCP in our implementation) with a congestion control (TFRC) guarantees the TCP-friendliness of our method, and TFRC makes it video streaming friendly.

Future works will include to implement and test other methods similar to VAAL, and to use also video quality metrics. Our final goal is to show that content adaptation on server application is the most appropriate video streaming method, not only in performance terms but also in implementation and practical terms, to cope with dynamic network bandwidth in cases such as video conferencing and small size video servers.

ACKNOWLEDGEMENTS

The authors thank Lotfi Amirouche, who wrote the first version of the DCCP client and server used for the experiments.

FUNDING

This work was supported by Ministry of High Education of Syria to WR.

REFERENCES

[1] Schulzrinne, H., Casner, S., Frederick, R., and Jacobson, V. (2003). RTP: A Transport Protocol for Real-Time Applications. RFC 3550, IETF.

[2] Kohler, E., Handley, M., and Floyd, S. (2006). Datagram Congestion Control Protocol (DCCP). RFC 4340, IETF.

[3] Floyd, S., Handley, M., Padhye, J., and Widmer, J. (2008). TCP Friendly Rate Control (TFRC): Protocol Specification. RFC 5348 (Proposed Standard), IETF.

[4] Ramadan, W., Dedu, E., Dhoutaut, D., and Bourgeois, J. (2011) RELD, RTT ECN Loss Differentiation to optimize the performance of transport protocols on wireless networks. *Telecommunications Systems, special issue on Mobile Computing and Networking Technologies*, **june**. Available online only.

[5] Su, Y.-C., Yang, C.-S., and Lee, C.-W. (2003) Optimal FEC assignment for scalable video transmission over burst error channel with loss rate feedback. *Signal Processing: Image Communication*, **18**, 537–547.

[6] Gürses, E., Akar, G. B., and Akar, N. (2005) A simple and effective mechanism for stored video streaming with TCP transport and server-side adaptive frame discard. *Computer Networks*, **48**, 489–501.

[7] Huszák, A. and Imre, S. (2006) Selective retransmission of MPEG video streams over IP networks. *International Symposium on Communication System Networks and Digital Signal Processing (CSNDSP)*, Patras, Greece, July 5, pp. 125–128.

[8] Kazantzidis, M. I. (2002) Adaptive Multimedia in Wireless IP Networks. PhD thesis University of California Los Angeles, USA.

[9] Haukaas, T. (2007) Rate adaptive video streaming over wireless networks. Master's thesis. Norwegian

University of Science and Technology Trondheim, Norway.

[10] Lie, A. and Klaue, J. (2008) Evalvid-RA: trace driven simulation of rate adaptive MPEG-4 VBR video. *Multimedia Systems*, **14**, 33–50.

[11] (2011). Information technology – dynamic adaptive streaming over HTTP (DASH) – part 1: Media presentation description and segment formats. ISO standard, Geneva, Switzerland. ISO/IEC 23009-1:2012.

[12] Liu, C., Bouazizi, I., and Gabbouj, M. (2011) Rate adaptation for adaptive HTTP streaming. *Second annual ACM conference on Multimedia systems*, New York, NY, USA MMSys, pp. 169–174. ACM.

[13] De Cicco, L. and Mascolo, S. (2010) An experimental investigation of the akamai adaptive video streaming. *6th international conference on HCI in work and learning, life and leisure: workgroup human-computer interaction and usability engineering*, Berlin USAB'10, pp. 447–464. Springer-Verlag.

[14] De Cicco, L., Mascolo, S., and Palmisano, V. (2011) Feedback control for adaptive live video streaming. *Second annual ACM conference on Multimedia systems*, New York, NY, USA MMSys, pp. 145–156. ACM.

[15] Feng, W. (2002) On the efficacy of quality, frame rate, and buffer management for video streaming across best-effort networks. *Journal of High Speed Networks*, **11**, 199–214.

[16] Balk, A., Maggiorini, D., Gerla, M., and Sanadidi, M. Y. (2003) Adaptive MPEG-4 video streaming with bandwidth estimation. *International Workshop on Quality of Service in Multiservice IP Networks*, London, UK, February 2, pp. 525–538. Springer-Verlag.

[17] Lee, S. and Chung, K. (2008) Buffer-driven adaptive video streaming with TCP-friendliness. *Computer Communications*, **31**, 2621–2630.

[18] Lee, S. and Chung, K. (2006) Mavis: Media-aware video streaming mechanism. *IFIP/IEEE International Conference on Management of Multimedia and Mobile Networks and Services (MMNS)*, Dublin, Ireland, October, pp. 74–84.

[19] Zambelli, A. (2009). IIS Smooth Streaming technical overview. Microsoft Corporation, Redmond, WA, US.

[20] (2010). Dynamic streaming in Flash Media Server 3.5. Available at `http://www.adobe.com/devnet/flashmediaserver/`. Adobe Systems Inc., California, US.

[21] Pantos, R. and May, W. (2011). HTTP live streaming. IETF Draft, available at `http://tools.ietf.org/html/draft-pantos-http-live-streaming-10`. IETF.

[22] ITU-T (2007). Opinion model for video-telephony applications. International Telecommunications Union, Geneva, Switzerland.

[23] Ramadan, W., Dedu, E., and Bourgeois, J. (2010) VAAL, video adaptation at application layer and experiments using DCCP. *WPMC 2010, 13th Int. Symposium on Wireless Personal Multimedia Communications*, Recife, Brazil, October, pp. 1–5. Springer. Proceedings on CD-ROM.

[24] Ramadan, W., Dedu, E., and Bourgeois, J. (2011) Avoiding zigzag quality switching in real content adaptive video streaming. *International Conference on Digital Information and Communication Technology and Its Applications (DICTAP)*, Dijon, France, June 1, pp. 421–435.

[25] Akhshabi, S., Begen, A. C., and Dovrolis, C. (2011) An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP. *Multimedia Systems*, San Jose, CA, US, February 2, pp. 157–168.

[26] McCanne, S., Jacobson, V., and Vetterli, M. (1996) Receiver-driven layered multicast. *SIGCOMM Computer Communication Review*, **26**, 117–130.

[27] Feamster, N., Bansal, D., and Balakrishnan, H. (2001) On the interactions between layered quality adaptation and congestion control for streaming video. *11th International Packet Video Workshop*, Kyongiu, Korea, April.

[28] Jain, R. K., Chiu, D.-M. W., and Have, W. R. (1984) A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. Research report TR-301. DEC, Littletown, MA, USA.

APPENDIX A. VAAL ALGORITHM

The classical code source for transmitting a statically-encoded video is the following:

```
while (not end of file)
  write video packet
  sleep
```

When doing adaptation with VAAL, this code is changed as follows:

```
for each period of time
  err = 0
  pkts = 0
  while (period not ended)
    pkts++
    write video packet
    if (write error)
       err++
    sleep
  errorRate = err/pkts
  if (errorRate == 0)
    increase bitrate
  else if (errorRate < 5%)
    maintain bitrate
  else
    decrease bitrate
```